Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2016) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016, 5-7 September 2016, York, United Kingdom

Effects of change propagation resulting from adaptive preprocessing in multicomponent predictive systems

Manuel Martin Salvador, Marcin Budka, Bogdan Gabrys

Data Science Institute, Bournemouth University. United Kingdom

Abstract

Predictive modelling is a complex process that requires a number of steps to transform raw data into predictions. Preprocessing of the input data is a key step in such process, and the selection of proper preprocessing methods is often a labour intensive task. Such methods are usually trained offline and their parameters remain fixed during the whole model deployment lifetime. However, preprocessing of non-stationary data streams is more challenging since the lack of adaptation of such preprocessing methods may degrade system performance. In addition, dependencies between different predictive system components make the adaptation process more challenging. In this paper we discuss the effects of change propagation resulting from using adaptive preprocessing in a Multicomponent Predictive System (MCPS). To highlight various issues we present four scenarios with different levels of adaptation. A number of experiments have been performed with a range of datasets to compare the prediction error in all four scenarios. Results show that well managed adaptation considerably improves the prediction performance. However, the model can become inconsistent if adaptation in one component is not correctly propagated throughout the rest of system components. Sometimes, such inconsistency may not cause an obvious deterioration in the system performance, therefore being difficult to detect. In some other cases it may even lead to a system failure as was observed in our experiments.

Peer-review under responsibility of KES International.

Keywords: Adaptive preprocessing; data streams; incremental learning

1. Introduction

In online supervised learning, data-driven predictive models are built incrementally using labelled instances. Preprocessing of those instances is often needed before the predictive model learning phase [11]. Common preprocessing methods include imputation of missing values, data normalisation, noise reduction and feature selection, to name a few. A combination of various preprocessing steps with a predictive model and possible postprocessing steps results in a so called multicomponent predictive system (MCPS [15]) which is the subject of this paper.

* Corresponding author.
E-mail address: msalvador@bournemouth.ac.uk

Peer-review under responsibility of KES International.

An MCPS, which converts raw data into predictions, can be represented as a WA-WF-net (Well-handled and Acyclic Workflow Petri net [15,17,19]). Formally,

$$MCPS = (P, T, F) \tag{1}$$

is a directed acyclic graph where P and T are finite sets of nodes (called places) and transitions, respectively, while F are the arcs connecting nodes. In an MCPS, a place can contain a single token which is represented as a tensor (i.e. multidimensional array). Further definition and properties of an MCPS are presented in [15].

The simplest MCPS includes a predictive model as a single transition (e.g. Naïve Bayes classifier). A more complex MCPS contains several preprocessing methods (e.g. moving average filter and PCA), an ensemble of predictive models (e.g. regression tree and neural network) and an ensemble aggregation method (e.g. majority voting). In this paper we have selected an intermediate example (see Fig. 1) consisting of three preprocessing methods (z-score normalisation, PCA, and min-max normalisation) and one classifier (GFMM neural network) as we explain further in Section 2. Please note that this example only serves as a convenient case of discussing the change propagation issues and other preprocessing steps and predictive models could have also been used.

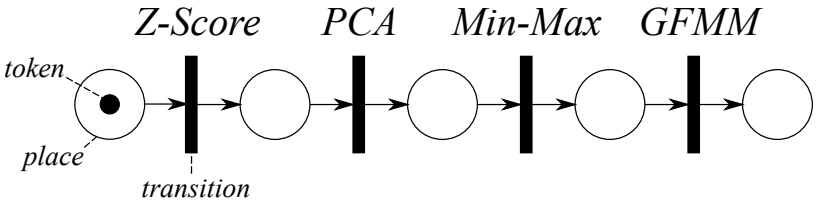


Fig. 1: Example of a Multicomponent Predictive System.

Preprocessing methods are usually trained offline and therefore their parameters remain fixed during the whole model lifetime. However, in non-stationary environments, where data statistics can vary over time, parameters of preprocessing methods should be updated to avoid performance loss. In addition, the predictive model may need to be adapted to new data as well.

There are two main types of strategies to adapt predictive models: (a) incremental update [5] or (b) replacement with a new model [9]. The GFMM model used as an example in this paper is updated using the former strategy, although other learning/adaptation strategies are also possible [6].

Adaptation of individual preprocessing methods has been considered by different authors. Adaptive feature selection [3], recursive PCA [13] and adaptive normalisation [16] are some examples of adaptive preprocessing methods. To the best of our knowledge, the work of Zliobaite and Gabrys [22] was the first attempt to address decoupling of adaptive preprocessing from the remaining components in a learning process with evolving data streams. One of the authors' conclusions was that such decoupling can be highly beneficial though can cause various problems if the change resulting from adapting a preprocessing step is not correctly propagated to subsequent components of an MCPS.

While in our previous research [14] we proposed autonomous adaptation strategies for the whole MCPS, in this work we concentrate on investigating and discussing the effects that an adaptation of a single preprocessing step can have on the overall predictive system behaviour, complexity and performance. We also introduce the concept of change propagation in MCPSs and highlight its importance when adapting individual components. Furthermore we show how the lack of a change propagation mechanism can lead to inconsistencies in the model representation.

The remainder of this paper has been organised as follows. Section 2 begins by introducing the adaptation of Multicomponent Predictive Systems, and presents the components of our case study. Section 3 describes four adaptation scenarios used in the experiments, followed by an experimental study with different datasets in Section 4. The paper concludes in Section 5.

2. Reactive Adaptation of Multicomponent Predictive Systems

In non-stationary data-streams, the relation between the input and the target can drift over time and predictive models have to be updated to avoid performance loss. Currently, the most common way to detect the need for

adaptation is to monitor model performance [8]. However, such an approach means that a number of instances have to be wrongly predicted before the problem can be actually detected. In addition, true labels (correct values which are predicted) are needed to estimate the loss and they may not always be available. In an MCPS we could add detection mechanisms in the early stages to monitor and adapt to the changes of the input data distribution (e.g. measuring abrupt changes in the mean of an input attribute) therefore not completely relying on the ability to calculate prediction error. We refer to this approach as reactive adaptation to the input data distribution changes.

A possible control mechanism is to establish preconditions and postconditions in all the system components. Therefore, if any of the conditions are not fulfilled, action must be taken. For example, in the case of min-max normalisation method if an input value is not within the $[min, max]$ range, the preprocessing component should be adapted to the new range. Other methods such as recursive PCA may include a post-condition that detects if previous and current principal components differ significantly.

In order to illustrate the point discussed above, we have selected three preprocessing methods (z-score normalisation, PCA for dimensionality reduction, and min-max normalisation) and one incremental predictive model (GFMM neural network). In the following subsections we briefly describe all four components.

2.1. Dimensionality reduction

Many datasets include variables which might not be relevant for the prediction task, unnecessarily increasing model complexity. For that reason, reducing the number of variables is often desirable. A common technique is performing Principal Component Analysis (PCA) and then selecting the first k components that explain a given percentage of dataset variance. Let $\mathbf{X}$ be the $n \times p$ input dataset where n is the number of samples and p the number of variables. Then, the reduced dataset is given by $\mathbf{X}' = \mathbf{X}\mathbf{W}_k$, where $\mathbf{W}_k$ is the $p \times k$ loading matrix.

2.2. Z-Score normalisation

Data normalisation is recommended before performing PCA, particularly if the variables have different units. Z-score normalisation, which scales the data to a standard normal distribution with zero mean and unit variance, is a common choice (see Eq. 2).

$$f(x) = \frac{x - \mu}{\sigma} \tag{2}$$

2.3. Min-max normalisation

The min-max method normalises the values of each attribute of a dataset to a range $[low, high]$ using the following formula, assuming $max > min$:

$$f(x) = (high - low) \cdot \frac{x - min}{max - min} + low \tag{3}$$

If the max and min values of an attribute are known a priori and do not change over time (e.g. they are given by the physical limitations of a measuring equipment), we can assume that all the values are going to be in the range $[low, high]$. On the other hand, if those limits are unknown, they have to be estimated from the available data.

Although we are aware of the consequences of not dealing with extreme outliers [18], we assume that they are being handled in a preceding preprocessing step for the sake of simplicity. For instance, Ogasawara et al. [16] smoothen the data and remove outliers during the training phase before the normalisation. Nevertheless, detection and handling of outliers are another preprocessing components that can potentially be adapted.

2.4. GFMM classifier

The General Fuzzy Min-Max (GFMM) neural network is a method that can be used for both classification and clustering problems. GFMM models the density of the input space using fuzzy hyperboxes (i.e. multidimensional generalisation of a rectangle with an associated membership function) to cover all training instances, with the j-th

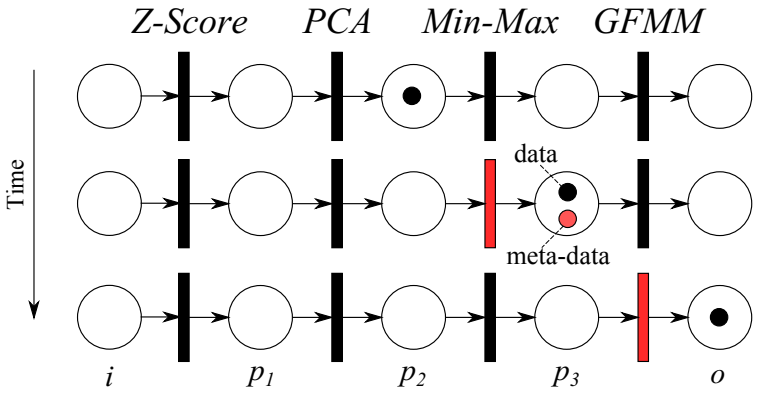


Fig. 2: Change propagation in an MCPS.

hyperbox being characterised by its min ($\boldsymbol{V}_j$) and max ($\boldsymbol{W}_j$) points (for an example refer to Fig. 4). The reason of selecting GFMM as model in this study is two-fold: 1) it is easy to interpret and visualise with two-dimensional data; and 2) it requires data within the $[0, 1]$ range so a min-max normalisation step is needed. For full description of the incremental learning algorithm and further details of the GFMM family, please refer to the relevant literature [7,6].

2.5. Change propagation

Change propagation is a recurrent topic in complex systems where changing a component may require to adapt further parts of the system [10]. Understanding how components are connected is crucial for performing a correct adaptation. Some works have addressed change propagation in different type of structures such as dynamic trees [2] or directed acyclic graphs [1]. However we are not aware of other works focusing on MCPSs.

Any parametric change in one component of an MCPS can have a knock-on effect on the data consumed by the remaining components. In this context, change propagation can be defined as the mechanism of transferring relevant information across the system to keep it consistent. For instance, if the input space is transformed, the model should be mapped to this new space. That is, any change in the preprocessing parameters transforming the input space should be reflected in the model. In our example, the three preprocessing methods perform consecutive linear transformations, so inverting such transformations should be done in the right order. In other cases however, such propagation may not be necessary.

Figure 2 illustrates how a change in the min-max component is propagated to the GFMM model via a coloured token with meta-data, using the Petri net formalism. The token triggers adaptation of GFMM using supplied information. Since our original formal definition of MCPS [15] does not support this kind of token, we therefore expand it and define a coloured MCPS (or cMCPS) as an MCPS with two additional conditions: a) the places $P \setminus \{i, o\}$ can contain up to two tokens assuming that each of them is of a different colour/type (i.e. data and meta-data) and, as a consequence, b) the Petri net is not safe (i.e. each place $p \in P$ can contain more than one token at any given time).

Although a sequence of linear transformations can be unified into a single linear transformation, the cost of system adaptation [21] can be reduced by inverting only the transformations of affected components. In our scenario, if *max* or *min* limits of Eq. 3 are updated, the hyperboxes have to be renormalised. To this end, the hyperboxes are first denormalised from the previous $[min', max']$ range using Eq. 4, and then normalised again with the new *max* and *min* values using Eq. 3.

$$f'(x) = \frac{x - low}{high - low} \cdot (max' - min') + min' \tag{4}$$

Similarly, if the loading vectors of PCA are modified, hyperboxes have to be denormalised using Eq. 4 before reversing the PCA transformation. The reverse PCA transformation is given by $\mathbf{X} = \mathbf{X}'\mathbf{W}_k^T$. Then, the hyperboxes are transformed using the new loading vectors and normalised again to the $[0, 1]$ range.

The same situation occurs when any parameters of z-score are changing (e.g. μ is now μ' and σ is σ'). All the transformations have to be reversed ($x' = \sigma x + \mu$) and then reapplied with the new parameters ($f(x') = \frac{x'-\mu'}{\sigma'}$). Otherwise, model representation would be inconsistent with the new input space.

While these reverse transformations may seem tedious and not always possible, in the following section we introduce four scenarios in order to investigate the severity of possible impact which may result from not propagating the changes correctly.

3. Scenarios

In this section we introduce four scenarios where data is preprocessed and then classified as shown in Fig. 1. All the scenarios share the same training phase, where a batch of data is preprocessed and then used to build the initial classifier. The main difference between the scenarios is the degree of adaptation of the predictive system, as summarised in Table 1:

#1 - No adaptation. All the components remain fixed at all times. The parameters extracted during the training phase are then used during online preprocessing of the data stream but no adaptation is performed at any level.

#2 - No adaptive preprocessing. The preprocessing methods are not adapted. However, this and the following scenarios assume that true labels of the instances are provided just after they are classified. Hence, the labelled instance is fed to the incremental learning algorithm that updates the classification model.

#3 - Adaptive preprocessing without propagation. A buffer with recent instances is used for computing the PCA loadings. Then, the incoming instances are transformed into the new space formed by the principal components. In addition, scaling parameters (max and min) are constantly checked and updated if any value is out of those limits. These local changes are however not propagated further through the model.

#4 - Adaptive preprocessing with propagation. Local changes are propagated through the system. The classification model is therefore mapped to the new input space: the hyperboxes of GFMM network are transformed to the original space by reversing the preprocessing steps, and then the updated preprocessing methods are applied to the hyperboxes.

Table 1: Summary table of scenarios.

Scenario	Adaptive Preprocessing	Change Propagation	Adaptive Model
#1	No	No	No
#2	No	No	Yes
#3	Yes	No	Yes
#4	Yes	Yes	Yes

4. Experimental study

This experimental study considers the scenarios presented in Section 3 in order to illustrate the implications of adapting some components of the preprocessing and the classifier.

To compare the system performance between the scenarios, the following values have been measured over four different datasets: a) classification error; b) maximum hyperbox membership degree (as a measure of classifier confidence); c) number of samples out of [0, 1] range; and d) number of hyperboxes.

The first 200 samples of each dataset have been selected for training and the rest for sequential testing/adaptation.

4.1. Synthetic data stream

A synthetic dataset (SYN) has been generated for illustration purposes, consisting of 2 classes that drift over time. The mean of attribute 1 is increasing over time for class 1, while the mean of attribute 2 remains constant. The opposite situation happens for class 2. Fig. 3 shows the state of the dataset at different timestamps.

The dataset contains 600 samples, 2 numerical attributes and 2 classes. Samples are randomly generated from a normal distribution with different means for each attribute but unit standard deviation. Only min-max normalisation is performed as part of preprocessing for the synthetic dataset.

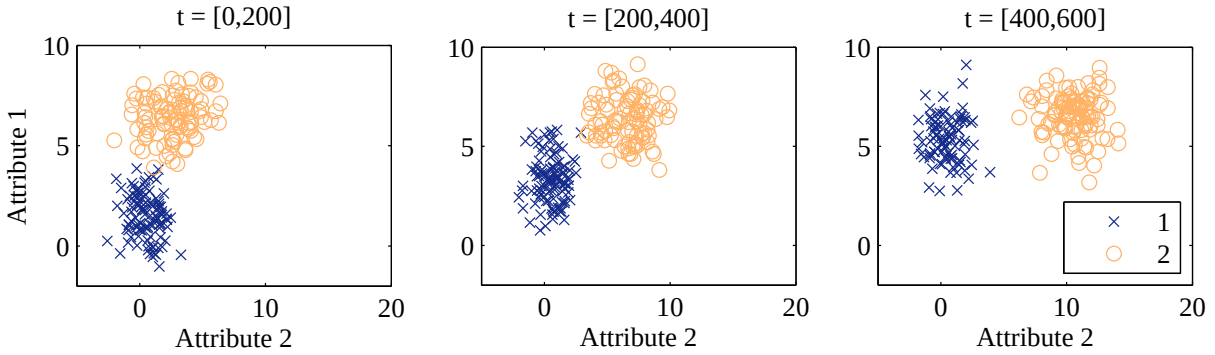


Fig. 3: SYN dataset at different timestamps.

4.2. Real data streams

Three publicly available datasets have been selected. The first real dataset (ELEC[12]) was collected from the Australian New South Wales Electricity Market and contains 45,312 instances, 7 attributes and 2 classes which are affected by demand and supply of the market.

The second dataset (COVERTYPE[4]) contains the forest cover type for 30x30 meter cells obtained from US Forest Service. The task is to predict forest cover type from cartographic variables. It consists of 581,012 instances, 54 attributes and 7 classes.

The third dataset (GAS[20]) contains 13,910 measurements from chemical sensors used to distinguish between different gases. It consists of 128 attributes and 6 classes.

For each dataset, the first 600 samples have been selected. First 200 samples have been used as training set for building the model and the rest for online testing. The preprocessing phase is composed of z-score normalisation, dimensionality reduction using PCA and finally, min-max normalisation.

4.3. Results for synthetic data stream

The results obtained from SYN dataset are shown in Figs. 4 and 5a. Fig. 4 shows the final snapshot of data points and hyperboxes for each scenario.

Points from class 1 (blue) are drifting to the space covered by a hyperbox of class 2 (gold) as can be seen in Fig. 4-a. This fact explains why the classification error is highest in Scenario 1, where the model is not updated. In Scenarios 2, 3 and 4 the model is evolving over time and hyperboxes are adjusted as shown in Figs. 4-(b,c,d), to capture new data points.

A direct consequence of scenarios without adaptive preprocessing (1 and 2) can be seen in Figs. 4-(a,b) where new points lie outside the $[0, 1]$ range and therefore cannot be classified by the predictive model since it only operates within the $[0, 1]$ range. Fig. 5a-c shows that the number of off-limit samples is increasing linearly over time because the data keeps drifting and the input space is not rescaled.

The adaptive normalisation carried out in Scenarios 3 and 4 guarantees that all the points are within the $[0, 1]$ range (see Figs. 4-c,d). Although the model of Scenario 3 has classified correctly most of the samples, it is actually

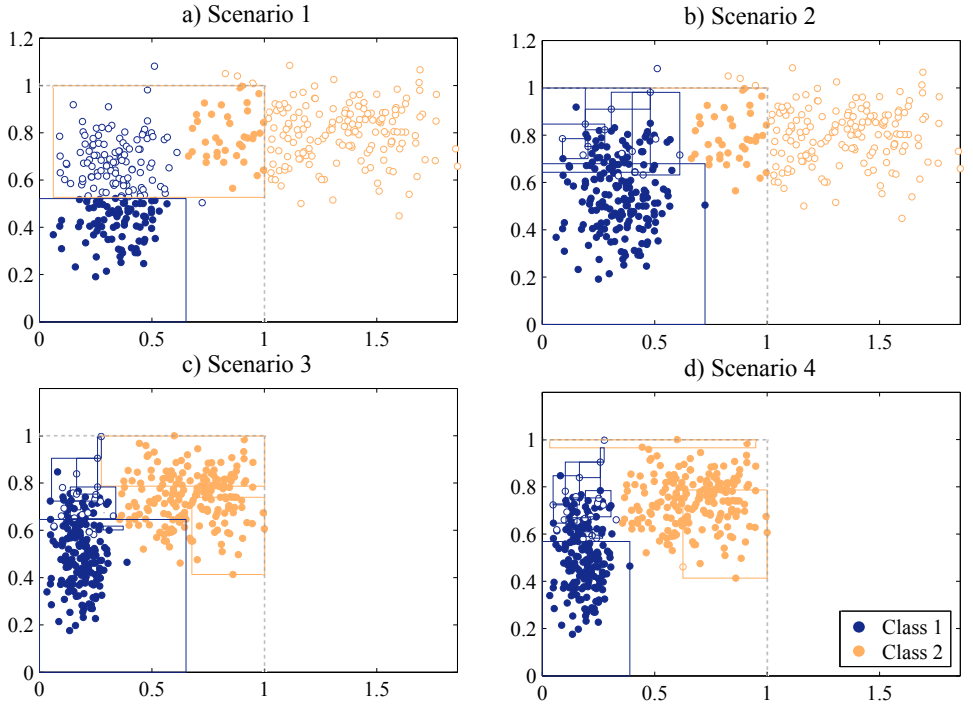


Fig. 4: Plots of the SYN testing data together with GFMM hyperboxes for each scenario.

inconsistent. That is, the rescaling factor is not propagated to the model and therefore its representation is incorrect. Such inconsistency is however not reflected in the classification results (see Fig. 5a-a) and hence might be impossible to detect. In contrast, the model representation in Scenario 4 is correct because the hyperboxes have been rescaled to the new input space (see Fig. 4-d).

The classification errors in Fig. 5a-a show that adapting the predictive system leads to considerably better results than in the static approach. Also the maximum membership degree is decreasing over time in Scenarios 1 and 2 (see Fig. 5a-b) which indicates a lower classification confidence. The lack of forgetting mechanism for GFMM method results in increase of model complexity over time, manifesting itself in growing number of hyperboxes (see Fig. 5a-d) and thus contributing to the model becoming more difficult to understand and interpret.

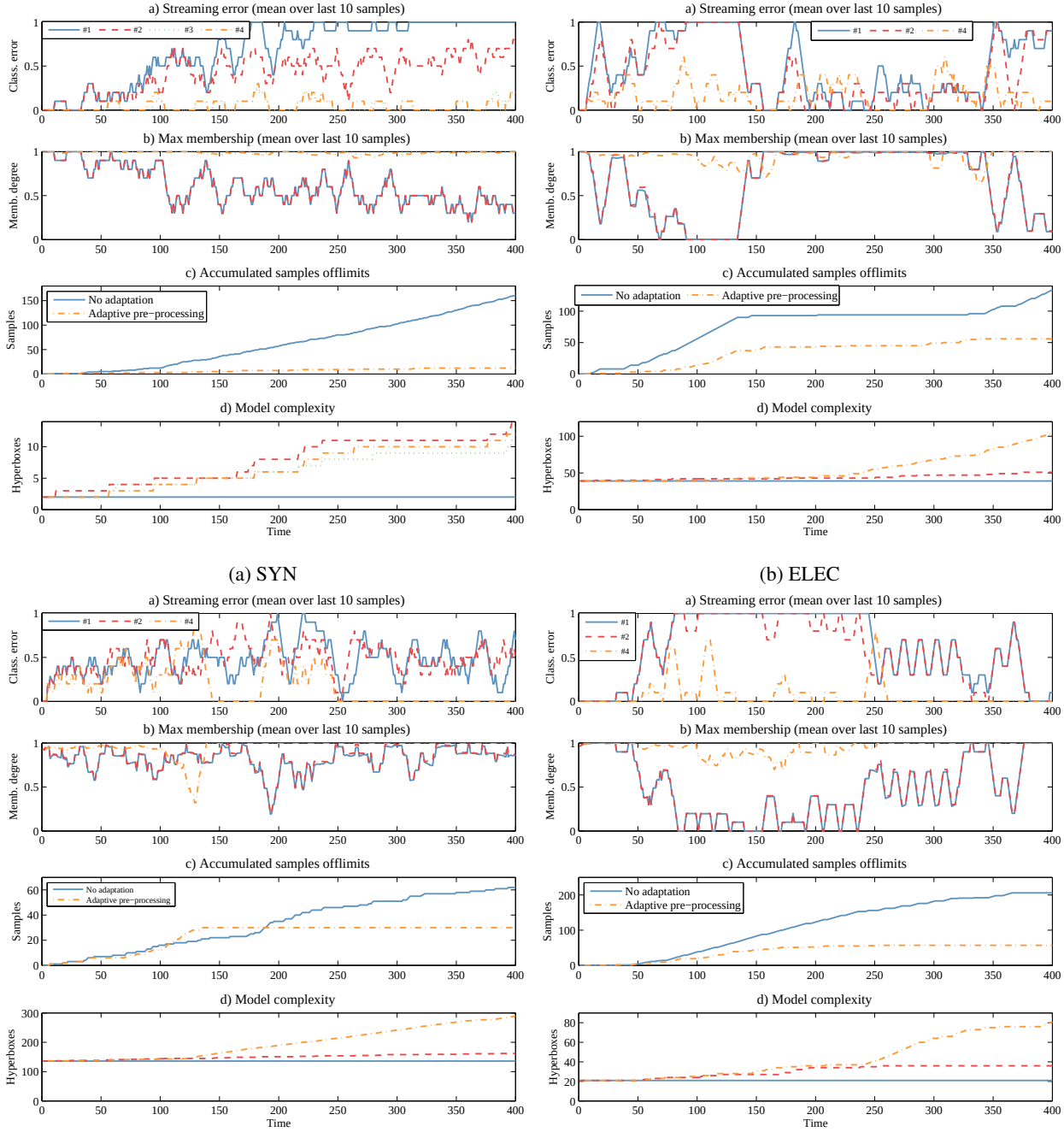
4.4. Results for real data streams

In general, Scenario 4 has a lower classification error in the three datasets (see Figs. 5b, 5c, and 5d). The classification error of Scenario 4 is considerably lower in the periods where there are new samples out of the min-max limits than in scenarios without adaptive preprocessing (see Figs. 5b, 5c, and 5d).

As observed in the synthetic dataset, there seems to be a negative correlation between the classification error and the maximum membership degree. Figs. 5b 5c, and 5d show that propagating the changes from the preprocessing methods to the model for adapting the hyperboxes increases the maximum membership degree and therefore the classification confidence. On the other hand, the fact of learning new concepts that cannot be classified by existing hyperboxes causes the creation of new hyperboxes that increases model complexity in Scenario 4 as shown in Figs. 5b, 5c, and 5d.

It is worth mentioning that the execution of Scenario 3 has failed in all three real datasets. That is, the lack of change propagation causes an inconsistency between the input data and the model when the number of principal components varies over time (see Fig. 6), leading to a runtime execution error. In such case it is easy to detect that something is going wrong. However, if the number of principal components is constant but the loading vectors actually change,

there is still an inconsistency between the input data and the model in Scenario 3 that will be more difficult to detect because it will not produce any software exception.



(a) SYN

(b) ELEC

(c) COVERTYPE

(d) GAS

Fig. 5: Results for SYN (top-left) and ELEC (top-right), COVERTYPE (bottom-left) and GAS (bottom-right).

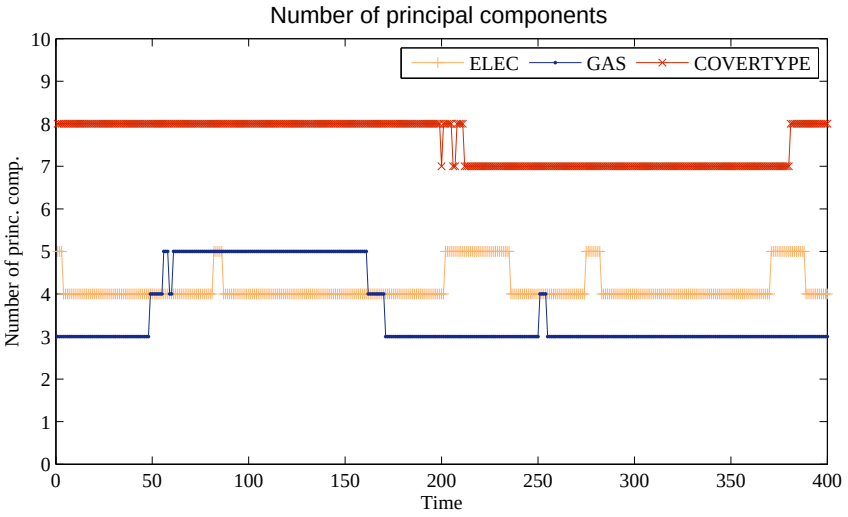


Fig. 6: Number of principal components of ELEC, GAS and COVERTYPE testing data.

Table 2: Accumulated classification error (%) for all datasets. Values in brackets denote contribution of offlimits samples to the error.

Dataset	Scenario #1	Scenario #2	Scenario #3	Scenario #4
SYN	69.25 (40.25)	43.75 (40.25)	4.75 (0)	5 (0)
ELEC	49.5 (33.5)	43.5 (33.5)	-	16.75 (0)
COVERTYPE	47.5 (15.5)	49 (15.5)	-	18.5 (0)
GAS	61.75 (51.5)	56.5 (51.5)	-	7.5 (0)

5. Conclusions

This study demonstrated that adapting individual components of an MCPS must be carried out very carefully taking into account the impact of a change on the rest of the system. In order to formalise the process of adapting components in the correct order, the concept of change propagation has been introduced. An illustrative example in which an MCPS consisting of three preprocessing methods and one classifier has been evaluated in four scenarios with different datasets.

The need of performing adaptive preprocessing is clear in all the tested datasets. The percentage of unclassified samples in scenarios without adaptive preprocessing varies between 15.5% and 51.5% (see Table 2). Consequently, the deterioration of system performance is considerable. A simple adaptive min-max normalisation method was shown to reduce the classification error when coupled with a classifier that requires such adaptation.

Modifying the input space requires however updating of model parameters. Otherwise, model representation can be inconsistent with the input data. Such inconsistency can even produce a system failure. To illustrate such adaptation, the GFMM method has been extended to react to such changes caused by the adaptive preprocessing methods. Other models will require similar type of transformations (e.g. changing node values in a decision tree).

While the paper represents an early stage of research and has been primarily intended to highlight a range of very important issues rather than provide comprehensive solutions, it has raised a number of questions in need of further investigation. Does a change in a component always need to be propagated? How to handle propagation in systems with nonlinear transformations? How to order the preprocessing methods to reduce the required propagation? How to reduce the cost of adaptation while maximising the benefits?

The above questions and the issues raised in this paper are particularly relevant and timely in the context of how easy it is to compose an MCPS in popular data analysis tools like KNIME, RapidMiner or SAS. All of these tools facilitate graphical composition of data transformation flows by linking data transformation blocks. While such ap-

proach provides a tremendous opportunity for non-expert users, as illustrated in this paper, it poses potential issues when MCPSs composed in such a way were to be deployed in non-stationary environments requiring adaptation and even more so if the preprocessing steps could be easily adapted but without awareness of a need for appropriate change propagation mechanisms currently not implemented or even available.

Acknowledgements

Research leading to these results has received funding from the European Commission within the Marie Curie Industry and Academia Partnerships and Pathways (IAPP) programme under grant agreement no. 251617.

References

1. U. A. Acar, G. E. Blelloch, and R. Harper. Adaptive functional programming. *ACM SIGPLAN Not.*, 37(1):247–259, 2002.
2. U. A. Acar, G. E. Blelloch, and J. L. Vittes. An experimental analysis of change propagation in dynamic trees. *Comput. Sci. Dep.*, 2005.
3. C. Anagnostopoulos, D. Tasoulis, D. J. Hand, and N. M. Adams. Online optimization for variable selection in data streams. In *Proceedings of the 2008 conference on ECAI 2008*, volume 1, pages 132–136. IOS Press Amsterdam, 2008.
4. J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999.
5. A. Bouchachia, B. Gabrys, and Z. Sahel. Overview of Some Incremental Learning Algorithms. In *2007 IEEE International Fuzzy Systems Conference*, number Il, pages 1–6. IEEE, 2007.
6. B. Gabrys. Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine? *Fuzzy Sets and Systems*, 147(1):39–56, 2004.
7. B. Gabrys and A. Bargiela. General fuzzy min-max neural network for clustering and classification. *IEEE transactions on neural networks*, 11(3):769–83, 2000.
8. J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with Drift Detection. In *Advances in Artificial Intelligence - SBIA 2004*, pages 286–295, 2004.
9. J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.
10. M. Giffin, O. de Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson. Change Propagation Analysis in Complex Technical Systems. *ASME J Mech Des.*, 131:1–14, 2009.
11. J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Elsevier, 3 edition, 2011.
12. M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.
13. W. Li, H. Yue, S. Valle-Cervantes, and S. Qin. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10(5):471–486, 2000.
14. M. Martin Salvador, M. Budka, and B. Gabrys. Adapting Multicomponent Predictive Systems using Hybrid Adaptation Strategies with Auto-WEKA in Process Industry. In *International Conference on Machine Learning. AutoML workshop (in press)*, 2016.
15. M. Martin Salvador, M. Budka, and B. Gabrys. Automatic composition and optimisation of multicomponent predictive systems. *IEEE Transactions on Knowledge and Data Engineering (under review - available at http://bit.ly/automatic-mcps-paper)*, 2016.
16. E. Ogasawara, L. C. Martinez, D. de Oliveira, G. Zimbrao, G. L. Pap, and M. Mattoso. Adaptive Normalization: A novel data normalization approach for non-stationary time series. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.
17. L. Ping, H. Hao, and L. Jian. On 1-soundness and Soundness of Workflow Nets. In *Third Work. Model. Objects, Components, Agents*, pages 21–36, 2004.
18. D. Pyle. *Data preparation for data mining*. Morgan Kaufmann, 1999.
19. W. M. P. Van Der Aalst. The Application of Petri Nets To Workflow Management. *J. Circuits, Syst. Comput.*, 08(01):21–66, 1998.
20. A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166-167:320–329, 2012.
21. I. Zliobaite, M. Budka, and F. Stahl. Towards cost-sensitive adaptation: When is it worth updating your predictive model? *Neurocomputing*, 2014.
22. I. Zliobaite and B. Gabrys. Adaptive Preprocessing for Streaming Data. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):309–321, 2014.